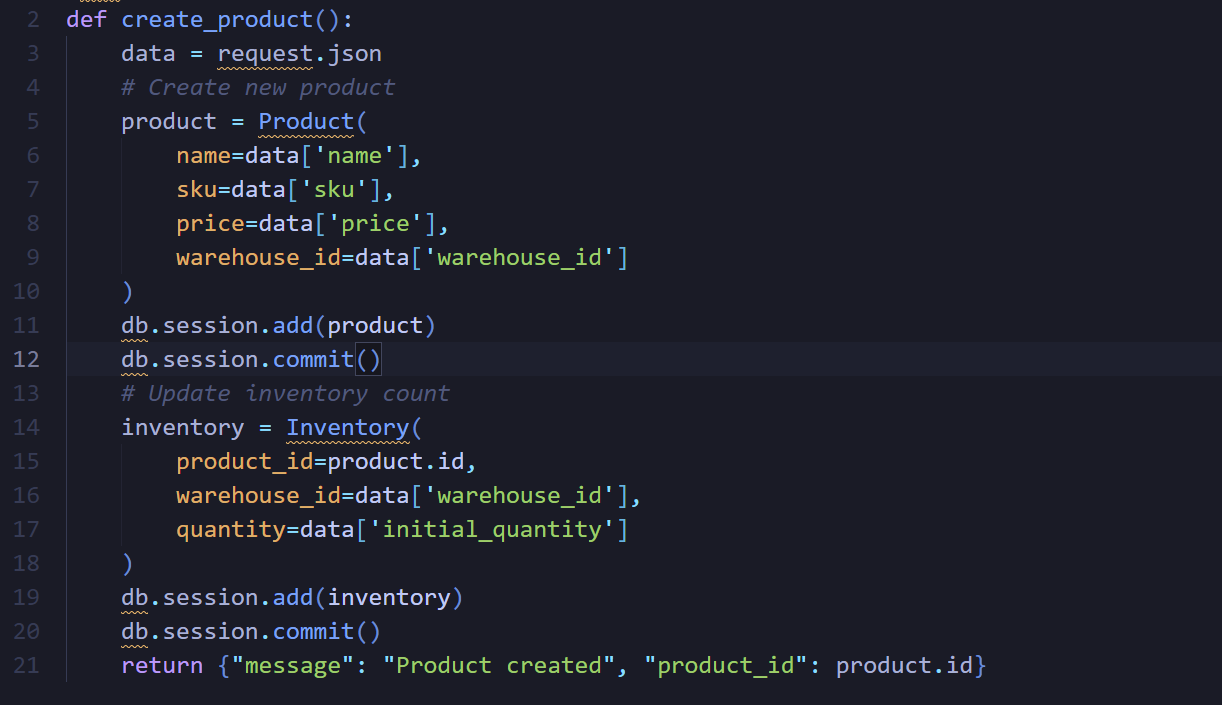
**Part 1: Code Review & Debugging**

**Problem 1 : Existence of all the required fields not being checked.**

The given code assumes that all the fields are present in the data.

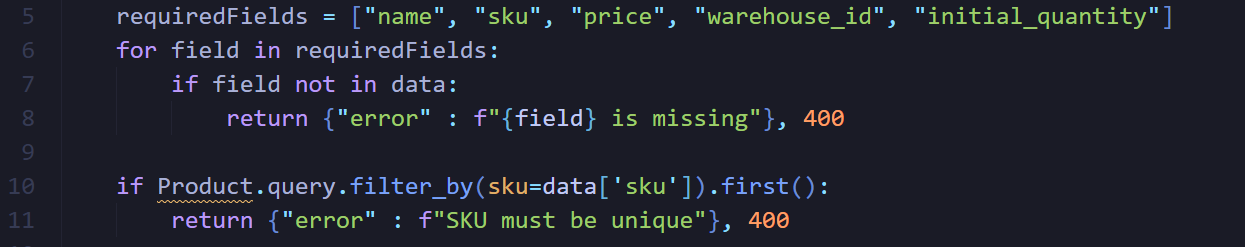


**Impact :**

In the case of a missing field, the API returns a 500 error with no explanation.

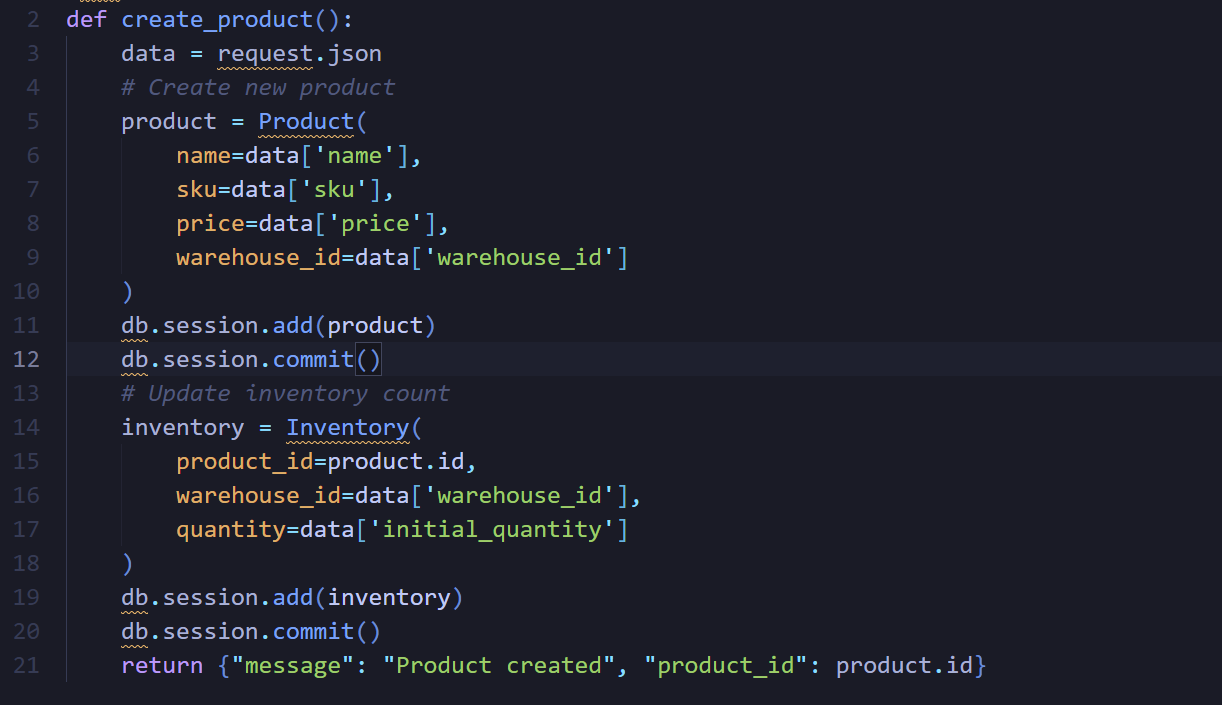
**Solution :**

A loop to check for missing fields and returns a 400 Bad Request with a clear error message.



**Problem 2 : SKU should be unique, no checks done if not.**

Products can be created with duplicate SKUs.

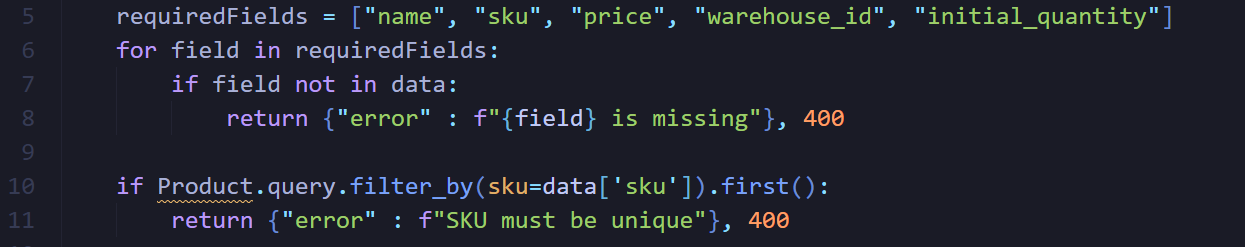


**Impact :**

Two products sharing the same SKUs will create confusions and break business rules.

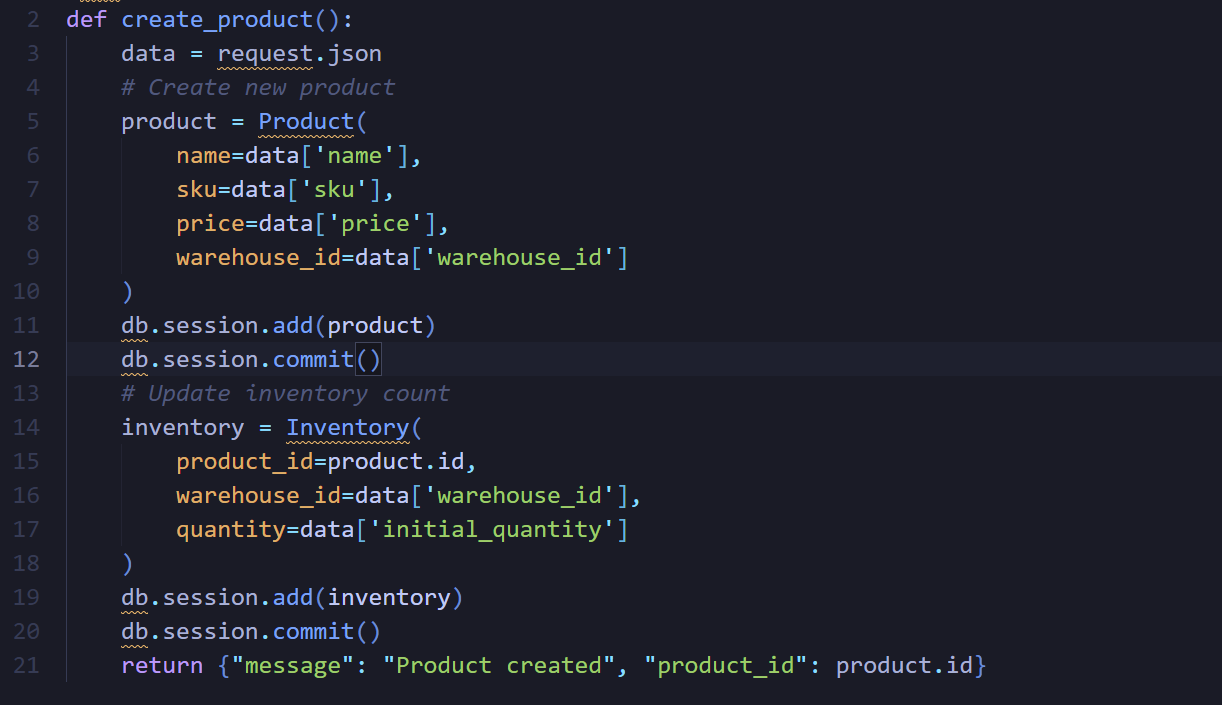
**Solution :**

Added a uniqueness check in the API.



**Problem 3 : Price stored as a float.**

Storing a currency as a float is not recommended as they lose precision.

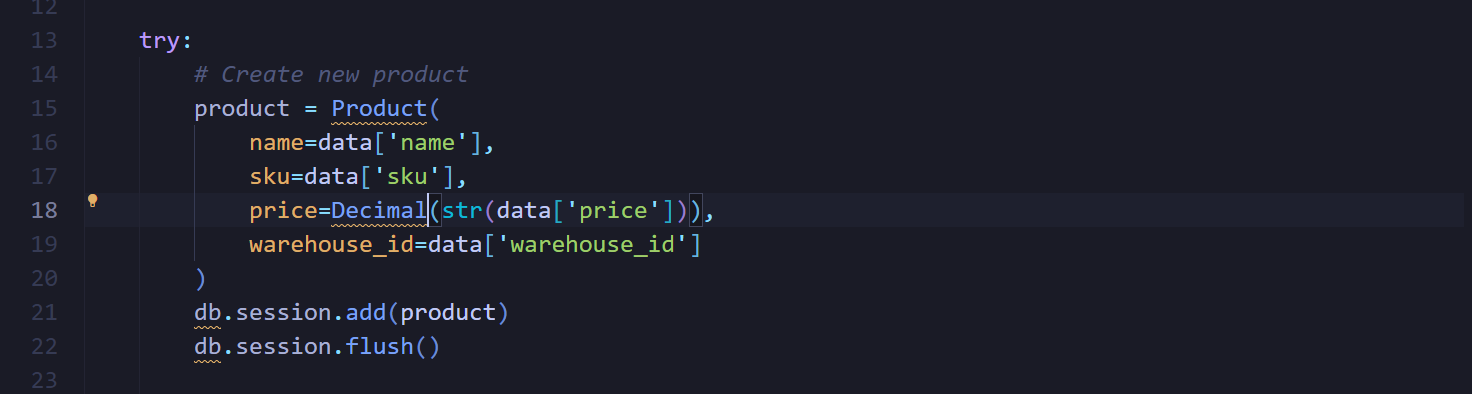


**Impact :**

Wrong totals while billing and inaccuracies in finances.

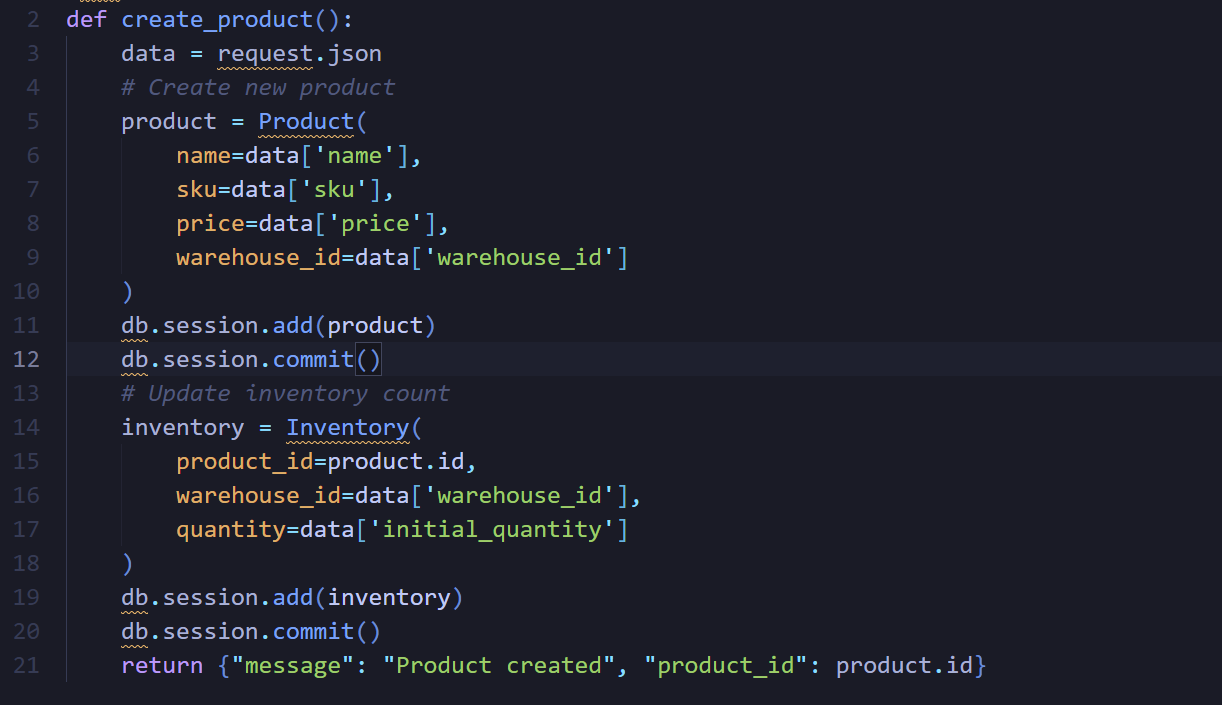
**Solution :**

The price is now stored as a Decimal which ensures precision.



**Problem 4 : Initial quantity is not optional.**

Code always expects “initial\_quantity”.

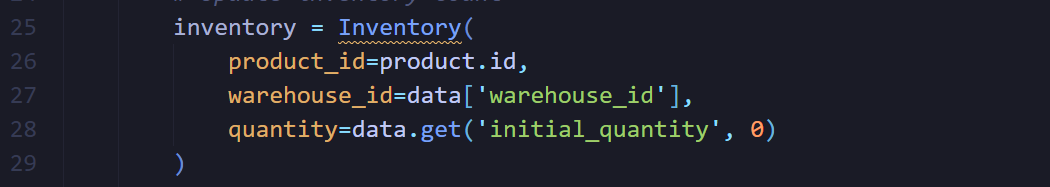


**Impact :**

If it is not provided, the API crashes.

**Solution :**

The API now defaults it to zero if not entered.



**Problem 5 : No error handling**

The code will crash if an error related to db occurs.

**Impact :**

API returns a 500 Bad Request with no clear exception.

**Solution :**

Try/except blocks added with db.session.rollback(). This ensures clear error response.

